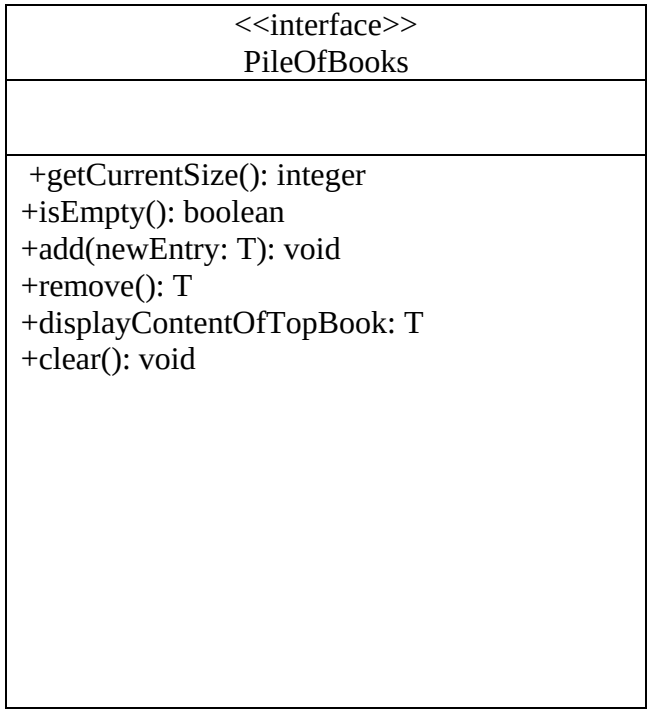
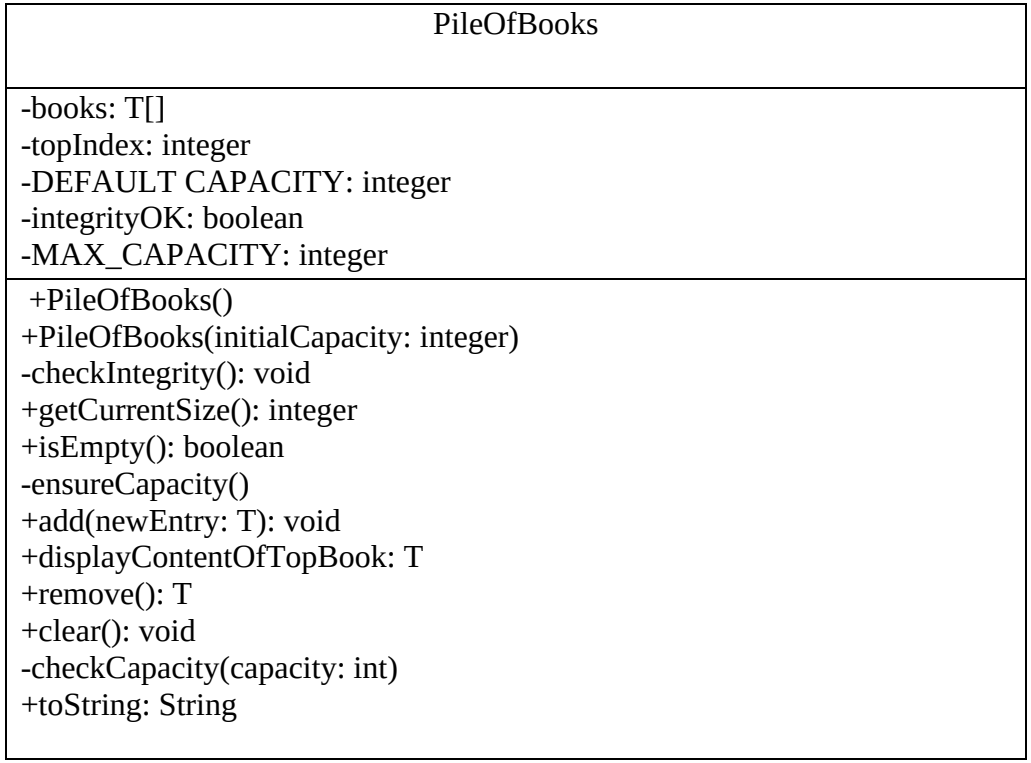
UML PileOfBooksInterface.java



UML PileOfBooks.java



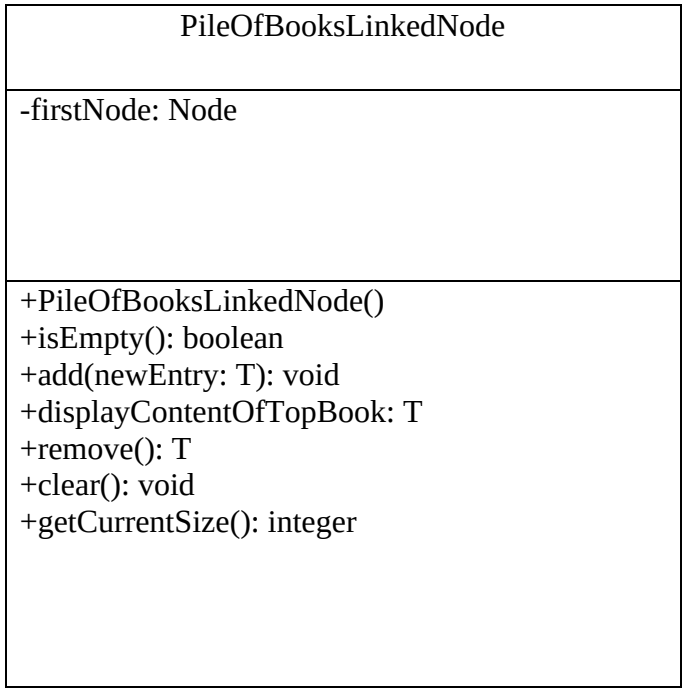
Taks 4

The PileOfBooks class **uses a resizable array**. According to the UML above, these are the analysis algorithm of each method in a PileOfBooks class.

Method	Time complexity	Description	Best case	Worst case
+getCurrentSize(): integer	O(1)	The operation getCurrentSize() can get the size of an array directly.		
+isEmpty(): boolean	O(1)	We only check to see if there is an element in a array, so this operation is fast.		
+add(newEntry: T): boolean	**O(1)**, O(n)	Adding a new entry after the last entry in the array of a pile of books. Therefore, a new added entry will be on the top index. Adding a new entry to the a pile of books is fast. Adding an element to a stack is almost O(1).	When ensureCapacity does not resize a pile of books, add a new entry is an O(1) operation, since its performance is independent of the size of the pile of books.	Resizing the array is an O(n) operation, so when the array is full, the performance of adding a new entry degrades to O(n).

+remove(): T	O(1)	Removing an unspecified entry is fast. Retrieve the top element, then perform a remove operation by decrement topIndex.		
+clear(): void	O(n)	It requires time to perform a remove operation for each entry. The more elements we have, the longer time needed to remove each entry.		
+displayContentOfTopBook: T	O(1)	Retrieve the top entry in the pile of books is fast.		

UML PileOfBooksLinkedNode.java



Taks 4

The PileOfBooksLinkedNode class uses a resizable array. According to the UML above, these are the analysis algorithm of each method in a PileOfBooksLinkedNode class.

Method	Time complexity	Description	Best case	Worst case
+getCurrentSize(): integer	O(1)	The operation getCurrentSize() can get the total elements in a chain directly.		
+isEmpty(): boolean	O(1)	It doesn't require to go through all element because we just set the topNode equals to null. This operation is fast.		
+add(newEntry: T): boolean	O(1)	This operation is independent of the other entries in the stack. Its performance is thus O(1).		
+remove(): T	O(1)	Removing an unspecified entry is fast. Getting the top entry in the stack, we can access the data portion of the first node in the chain.		
+clear(): void	O(1)	The same scenario as isEmpty() in linked node. It doesn't require to go through all		

		element because we just set the topNode equals to null. This operation is fast.		
+displayContentOfTopBook: T	O(1)	The same scenario as remove operation. Getting the top entry in the stack, we can access the data portion of the first node in the chain.		